METODER KI ASSIGNMENT 3

Exercise 1

Unobserved and observed variables:

The unobserved variable X_t for the umbrella world is wether it is raining at time slice t or not.
The observed variable E_t is wether or not the director brought an umbrella at a specific time slice.

Dynamics as matrices:

$$P(X_t \mid X_{t-1}) \qquad P(E_t \mid X_t)$$

$$\begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \qquad \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$

Assumptions in model:

One assumption that is made in this model is that wether it rains or not tomorrow depends only on wether or not it rained today. In a Markov model we assume the state variables contain all the necessary values to determine the probability for the next time slice. In this model we could improve the model in many different ways, for example by transforming it into second order and calculating probability based on the two previous time slices. We could also increase the set of variables in which the probabilities are based upon, for example by adding which season we are currently in.

Exercise 2

Code:

```python
import numpy as np

def run_HMM_filter_forward(initial_state, transition_matrix, emission_matrix, observations):
    forward = np.zeros((len(initial_state), len(observations)))
    forward[:, 0] = initial_state * emission_matrix[:, observations[0]]
    for i in range(1, len(observations)):
        forward[:, i] = np.dot(forward[:, i-1], transition_matrix) * emission_matrix[:, observations[i]]
    prob = forward[:, -1] / np.sum(forward[:, -1])
    return prob


def main():
    initial_state = np.array([0.5, 0.5])
    transition_matrix = np.array([[0.7, 0.3], [0.3, 0.7]])
    emission_matrix = np.array([[0.9, 0.1], [0.2, 0.8]])
    observations = np.array([0, 0, 1, 0, 0])
    prob = run_HMM_filter_forward(initial_state, transition_matrix, emission_matrix, observations)
    print(f"\nInitial state: \n{initial_state}")
    print(f"\nTransition matrix: \n{transition_matrix}")
    print(f"\nEmission matrix: \n{emission_matrix}")
    print(f"\nObservations: {observations}")
    print(f"\nP(rain|observations) = {prob[0]:.4f}")
    print(f"P(sun|observations) = {prob[1]:.4f}\n")

if __name__ == "__main__":
    main()
```

Part 1:

```
Initial state:
[0.5 0.5]

Transition matrix:
[[0.7 0.3]
 [0.3 0.7]]

Emission matrix:
[[0.9 0.1]
 [0.2 0.8]]

Observations: [0 0]

P(rain|observations) = 0.8834
P(sun|observations) = 0.1166
```

Part 2:

```
Initial state:
[0.5 0.5]

Transition matrix:
[[0.7 0.3]
 [0.3 0.7]]

Emission matrix:
[[0.9 0.1]
 [0.2 0.8]]

Observations: [0 0 1 0 0]

P(rain|observations) = 0.8673
P(sun|observations) = 0.1327
```